BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　昭64－72237

⑤1Int.Cl.⁴	識別記号	庁内整理番号
G 06 F 9/32	320	D－7361－5B
9/26	320	A－7361－5B

⑬公開　昭和64年(1989)3月17日

審査請求　未請求　発明の数 1　(全7頁)

⑤4発明の名称　アドレス計算方式

⑳特　願　昭62－228337

㉒出　願　昭62(1987)9月14日

⑦2発明者　海永　正博　神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

⑦2発明者　橋本　幸治　東京都小平市上水本町1479番地　日立マイクロコンピュータエンジニアリング株式会社内

⑦1出願人　株式会社日立製作所　東京都千代田区神田駿河台4丁目6番地

⑦1出願人　日立マイクロコンピュータエンジニアリング株式会社　東京都小平市上水本町1479番地

⑦1出願人　坂村　健　東京都港区白金台3－12－30－105

⑦4代理人　弁理士　小川　勝男　外1名

最終頁に続く

明　細　書

1．発明の名称

アドレス計算方式

2．特許請求の範囲

(1) 第1類アドレス指定フィールドで指定された値を計算途中アドレスの初期値とし、連続的に配置された複数の第2類アドレス指定フィールドの指示に従い順次に計算途中アドレス計算し、最終アドレスを求めることを特徴とするアドレス計算方式。

(2) 第1項記載のアドレス計算方式において、第2類アドレス指定フィールドの指定によりアドレス加算の1つのオペランドとしてプログラムカウンタの内容を指定できることを特徴とするアドレス計算方式

(3) 第1項記載のアドレス計算方式において、第2類アドレス指定フィールドの指定により計算したアドレスのロケーション内容を新たな計算途中アドレスとする場合に、該ロケーションのサイズを情報処理装置の語サイズの半分または4分の1に指定できることを特徴とするアドレス計算方式。

3．発明の詳細な説明

[産業上の利用分野]

本発明は情報処理装置において、機械命令オペランドのアドレス計算方式にかかわる。

[従来の技術]

情報処理装置の機械命令として、無条件分岐命令(JMP命令)というものがある。これはオペランドで指定されたアドレスへの無条件分岐を引き起こす命令である。

無条件分岐命令では、1つのアドレスにしか分岐できないが、多方向分岐命令を備えた情報処理装置も知られている。たとえば、共立出版社発行のVAX-11 アーキテクチャ ハンドブックによれば、VAX-11は多方向分岐命令(CASE命令)を提供している。この命令機能は以下の通りである。

まず使用例は以下のようになる。ここで、disp[n]は各々2バイトサイズである。

```
CASE selector,base,limit
    disp[0]
    ...
    disp[limit]
```

そして、この命令発行時の動作は第２図のマイクロプログラムにより定義される。第２図において個別の動作は、プログラミング言語Ｃに準じて、いわゆる高級言語の水準で記述してあるが、本来はハードウエアの個別ゲートを制御するいわゆる低水準の処理に分解されよう。しかし、ここでは、いたずらに処理内容を詳細化し、それにより本発明の本質を理解しずらくすることを避けるために、また個別ゲートの制御自体は本発明の範囲外であるので、第２図で記述された水準の処理ステップで、この動作を説明することにする。

第２図において、Ｔはある作業レジスタ、selector,base,limitは命令オペランドを受ける作業レジスタ、PCはプログラムカウンタである。また'<='は比較(小、または一致)演算子、disp[T]は第Ｔ番目のdisp要素の参照、'<<'は左シフト演算子をそれぞれ表現する。尚、VAX-11のCASE命令実行時のプログラムカウンタPCの値は最初のdisp要素のアドレスを指している。

命令デコーダが命令ストリーム中にCASE命令を検出すると、命令デコーダは第２図に示す処理フローのマイクロプログラムを起動する。まずステップ２０１において、selectorの値からbaseの値を減算し、結果の値を一時レジスタＴにセットする。次にステップ２０２において、一時レジスタＴの値とlimitの値を比較する。そして、Ｔの値がlimitの値以下であれば、次にステップ２０３において、PCの値とdispの第Ｔ番目の内容を加算し、その結果の値でPCを更新し、そしてステップ２０５にジャンプする。一方、Ｔの値がlimitの値以下でなければ、ステップ２０４にジャンプし、ステップ２０４において、PCの値とlimitの値を１ビット左シフトした値を加算し、その結果の値でPCを更新し、ステップ２０５に進む。そして、ステップ２０５において、'exit()'によりマイクロプログラム処理の終了を命令デコーダに通知することにより、CASE命令処理が終了する。

尚、ステップ２０３を経由することは、dispの第Ｔ(selector-base)番目の内容にもとずいて多方向分岐することに他ならない。一方ステップ２０４を経由することは、Ｔ(selector-base)の値が望まれた範囲外であって、従ってdisp配列の直後のアドレスに抜けることに他ならない。

以上のようにして、CASE命令を用いて多方向分岐を実現できる。

［発明が解決しようとする問題点］

以上示したように、CASE命令を用いると多方向に分岐できてなるほど便利である。しかし、この命令では分岐しかできない。多方向分岐の他に、多方向メモリアクセスというものも考えられる筈であるが、CASE命令では多方向分岐しかできない訳である。

この対策として、個別の命令、例えばメモリ内容反転命令(NEG命令としておく)に対応して、多方向反転命令(CASE_NEG命令としておく)を用意するという対策もありえるが、この対策は命令の数が増えすぎ、現実的な対策とはいえない。

本発明の目的は、JMP命令で多方向分岐を実現できるといった柔軟なアドレス計算方式を実現することにある。

［問題点を解決するための手段］

そこで、本発明では、CASE命令のアドレス計算部分をCASE命令の機能から分離させ、そしてオペランドのアドレス計算の機能に含ませることにした。

［作用］

これにより、CASE命令は不要となった。というのは、JMP命令の分岐アドレス指定において、CASE命令の機能から分離されたCASE命令のアドレス計算部分を指定するようにすれば良いからである。

勿論、本発明により、CASE_NEG命令の機能はNEG命令で実現できる。

［実施例］

以下、本発明の１実施例を詳細に説明する。

第５図は本発明を実施した情報処理装置の構成図である。主メモリ５１０には機械語プログラム５１１が格納されており、またデータ領域５１２が配置される。ＣＰＵ５２０は主メモリにアクセスしつつ機械語プログラム５１１の指示に従いアクセスすべきアドレスを計算しデータ領域５１２の値を参照、更新することによりデータ処理を遂行する。さらに機械語プログラム５１１の指示によっては、入出力装置５３０を介して情報処理装置と外界とのやりとりが行われる。読みだし専用メモリ５２１内にはマイクロプログラムが格納される。このマイクロプログラムは、機械語の指示などを解釈、実行するためのプログラムである。

第３図はオペランドを１つ持つ命令の本実施例による形式を表現している。フィールド３０１は命令コードが格納される部分である。フィールド３０２はオペランドの指定法が格納される部分である(第１類アドレス指定フィールド)。オペランドの指定法によっては、後続の位置に追加フィールド(第２類アドレス指定フィールド)が配置される。以下、本発明の説明に必要な範囲で、オペランド指定法を示しておく。

ﾌｨｰﾙﾄﾞ２		意味
0001	n	第ｎ汎用レジスタをオペランドとする
0000	1111	PCの値を途中アドレス(tmp)の初期値とし、追加フィールドの指定に従いアドレス計算

追加フィールドの構造は第４図の通りであり、個別のフィールドの意味は以下の通りである。尚、計算途中アドレスをtmpで表現する。また、indexという変数を導入する。

disp:オフセット(ｉフィールド参照)

xx　:シフトカウンタ(ｉフィールド参照)

n　:汎用レジスタ番号(ｉフィールド参照)

m　:インデックス指定子

0:nﾌｨｰﾙﾄﾞで指定した汎用ﾚｼﾞｽﾀ

index= GRn

1:　index= PC

i　:間接指定子

0:間接参照なし

tmp=　tmp+(index<<xx)+disp

1:間接参照あり

tmp= *(tmp+(index<<xx)+disp)

s　:サイズ指定子

0:間接参照時、4nﾊﾞｲﾄﾛｹｰｼｮﾝ

1:間接参照時、2nﾊﾞｲﾄﾛｹｰｼｮﾝ

e　:継続指定子

0:アドレス計算終了

1:次の追加フィールドのアドレス計算に進入

さて、この追加フィールドを具体的に解釈実行するマイクロプログラムを第１図に示す。この図で、curpは現追加フィールド指し示す作業レジスタとし、命令デコーダが正しく値をセットしているものとする。PCは現在実行中の命令アドレスを指すいわゆるプログラムカウンタである。GRnはcurpが指す追加フィールド内の部分フィールドnで特定される汎用レジスタである。'->'はポインタオフセット演算子であり、例えば'curp->m'でcurpが指す追加フィールド内m部分フィールドの参照を表現する。'?'、':'の２つで条件評価演算子となり、例えば'(curp->m==1)?PC:GRn'で、curp->mの値が１か検査し、そうであればPCの内容を式の値とし、そうでなければGRnの内容を式の値とすることを表現する。'*(short*)'、'*(int*)'で間接参照を表現している。例えば'*(short*)tmp'でtempの値が指している２バイトメモリロケーションへの参照を表現し、同様に'*(int*)tmp'でtempの値が指している４バイトメモリロケーションへの参照を表現している。また、'++'は＋１演算子であり、例えば'curp++'でcurpが次の追加フィールドを指すことになる。

第２図のマイクロプログラムソーステキストと同じく、この第１図のマイクロプログラムソーステキストをコンパイルまたはアセンブルして0/1のビットパタンで構成される２進のマイ

クロプログラムとし、これを第５図のマイクロプログラム格納用の読みだし専用メモリ５２１に格納すれば、本発明が情報処理装置に実際に組み込まれることになるのは勿論である。尚、第１図のフローチャート表現が第６図となる。

さて、本発明によって、JMP命令で多方向分岐が効率的に行えることを示すためにJMP命令とそれに関連した情報が以下の形でメモリロケーションに配置されているものとする。そして、汎用レジスタGR0の値に応じて、L0,L1,..,L10のアドレスに多方向分岐するものとする。尚GR0の値の範囲は０～１０が保証されているものとする。

Lb:	JMP	00001111
	第１追加ﾌｨｰﾙﾄﾞ	
	第２追加ﾌｨｰﾙﾄﾞ	
	L0-Lb	
	L1-Lb	
	L10-Lb	

また、２つの追加フィールドの詳細は以下の通りとする。

ei		n		msxx		disp
1	1	0	0	0	1	L0-Lb
0	0	0	1	0	0	0

第１追加フィールド
第２追加フィールド

この２つの追加フィールドが定めたtmp計算は以下の通りとなろう。

第１追加ﾌｨｰﾙﾄﾞ：tmp= *(tmp+(GR0<<1)+L0-Lb))
第２追加ﾌｨｰﾙﾄﾞ：tmp= tmp+(PC<<0)+0

さて、命令デコーダがアドレスLbのJMP命令を検出したものとする。このとき、命令デコーダはまず、オペランドのアドレス計算を行う必要がある。そして、今の場合オペランドフィールドのパタンが'00001111'であり、従って、作業レジスタtmpにPCの値をセットして、第１図のマイクロプログラムを起動する。ステップ１０１の検査は常に成功終了して、従って次にステップ１０２に進む。ステップ１０２では現追加フィールドのmサブフィールドの値が１か検査し、今の場合１ではなく、さらに現追加フィールドのnサブフィールドの値が０であるので、GR0の値を作業レジスタindexにセットする。次にステップ１０３で、tmpの値と、indexの値を現追加フィールドのxxサブフィールドの値(実は１)で左シフトした値と、現追加フィールドのdispサブフィールドの値(実はL0-Lb)の３者を加算し、その結果の値をtmpに、セットする。このとき、tmpの元々の値はLbであり、またdispはL0-Lbであり、従ってtmp+dispは実はL0である。従って、ステップ１０３の結果tmpにセットされた値は、じつは第２追加フィールド以降に配置されたオフセット配列のGR0の値番目のアドレスを指している。次にステップ１０４で、現追加フィールドのiサブフィールドの値が１であるか検査し、今の場合１でないので、ステップ１０５に抜ける。ステップ１０５では、現追加フィールドのsサブフィールドの値が１であるか検査し、今の場合１であるので、tmpが指す２バイトロケーションの内容を読みだし、それをtmpにセットする。次にステップ１０６で、現追加フィールドのeサブフィールドの値が０であるか検査し、今の場合0でないので、ステップ１０７に抜ける。ステップ１０７で、次の追加フィールドをcurpが指すように更新し、そしてステップ１０１にジャンプすることにより、次のループ処理を開始する。

ステップ１０１の検査は常に成功終了して、従って次にステップ１０２に進む。ステップ１０２では現追加フィールドのmサブフィールドの値が１か検査し、今の場合１であるので、PCの値を作業レジスタindexにセットする。次にステップ１０３で、tmpの値と、indexの値を現追加フィールドのxxサブフィールドの値(実は０)で左シフトした値と、現追加フィールドのdispサブフィールドの値(実は０)の３者を加算し、その結果の値をtmpにセットする。このとき、tmpの元々の値は実は第２追加フィールド以降に配置されたオフセット配列のGR0の値番目のロケーション内容であり、従って、ステップ１０３の結果tmpにセットされた値は、実はL

0,L1,..L10の内のGR0で特定されるアドレスということになる。次にステップ１０４で、現追加フィールドのiサブフィールドの値が１であるか検査し、今の　合１であるので、ステップ１０６にジャンプする。次にステップ１０６で、現追加フィールドのeサブフィールドの値が０であるか検査し、今の場合0でないので、'exit()'によりアドレス計算処理を終了し、その旨命令デコーダに通知する。通知を受けた命令デコーダは、今度は命令処理を遂行する必要がある。本実施例における命令デコーダは分岐命令の命令処理を自分自身で行うとする。この場合tmpの値を命令デコーダ自身がプログラムカウンタPCにセットすることにより、JMP命令の命令処理が終了することになる。

さて、この新PCはL0,L1,..L10の内のGR0で特定されるアドレスであり、従ってJMP命令により多方向分岐命令の機能が実現できたことになる。

［発明の効果］

以上示したように本発明により、JMP命令で多方向分岐命令の機能を実現できる。またJMP命令の代わりに、NEG命令を用いれば勿論CASE_NEG命令の機能を実現できることも明らかである。

本実施例の説明において、第２の追加フィールドの指定により、PCの値をアドレス計算に用いた。このときのPCの値は、最終アドレスを求めるためのベースアドレスとして用いられているものと解釈できる。そして、第１の追加フィールドの指定により求めたtmpの値は、ベースアドレスからのオフセットとして解釈できる。勿論、第２追加フィールドの後続の位置に配置された配列の個別要素は、本実施例の説明においては、まさしくL10-L0などのオフセットが配置されていたのであったが、オフセットであれば４バイトサイズである必要はなく、例えば本実施例でのように２バイトサイズでもよく、また１バイトサイズであっても構わない。そして、オフセット配列の要素数が多い状況であれば、要素サイズを小さくすることにより、全体のメモリサイズを小さくできることになる。

４．図面の簡単な説明

第１図は本発明の実施例によるアドレス計算のマイクロプログラムフローを示す図、第２図は従来技術によるCASE命令機能の実現例を示す図、第３図は本発明の実施例で想定した１オペランド命令のフィールド分割図、第４図は本発明によるアドレス計算用追加フィールドのサブフィールド分割図、第５図は本実施例による情報処理装置の構成図、第６図は第１図を所謂フローチャートで表現した図である。

第 1 図

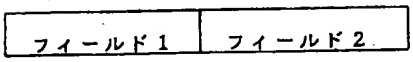
```
while(1){                                        ---101
   index=(curp->m==1)?PC:GRn;                    ---102
   tmp=tmp+(index<<(curp->xx))+disp;              ---103
   if(curp->i==0) goto L;                         ---104
   tmp =(curp->s==1)?                             ---105
         *(short*)tmp: *(int*)tmp;
 L:if(curp->e==0) exit();                         ---106
   curp++;                                        ---107
}
```

代理人弁理士 小川勝男

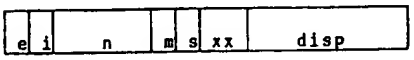
第 2 図

```
T = selector-base;          -----201
if( T <= limit)             -----202
   PC = PC + disp[T];       -----203
else
   PC = PC + (limit<<1);    -----204
exit();                     -----205
```

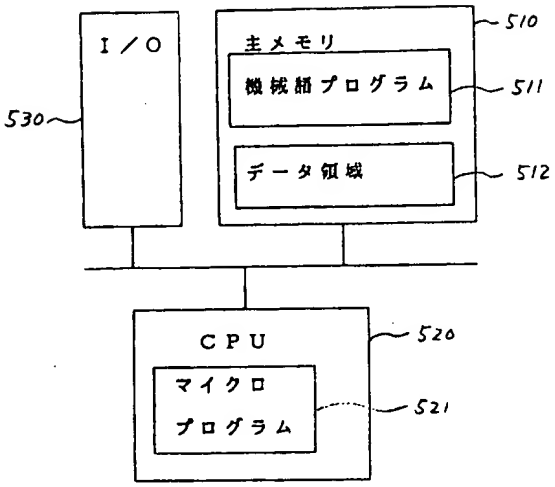
第 3 図



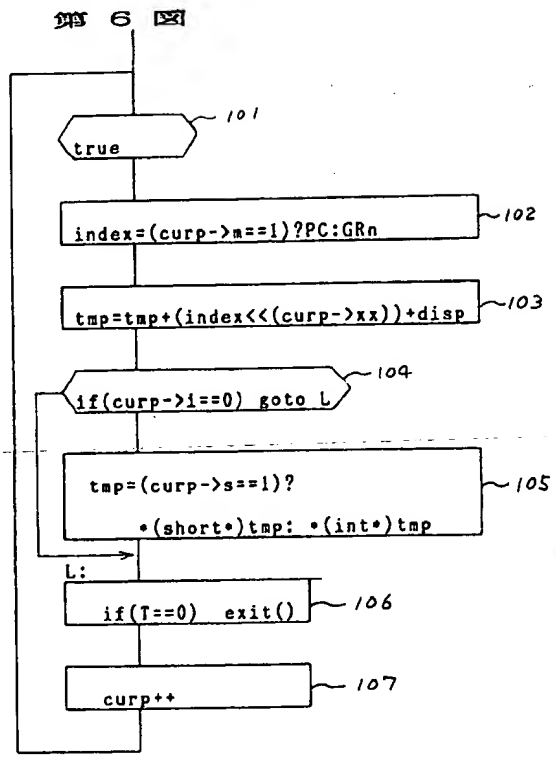
第 4 図



第 5 図



第 6 図



第1頁の続き

⑫発明者　川崎　郁也　東京都小平市上水本町1450番地　株式会社日立製作所武蔵工場内

⑫発明者　長谷川　淳　東京都小平市上水本町1479番地　日立マイクロコンピュータエンジニアリング株式会社内

⑫発明者　坂村　健　東京都港区白金台3－12－30－105

⑫発明者　秋田　英彦　神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内